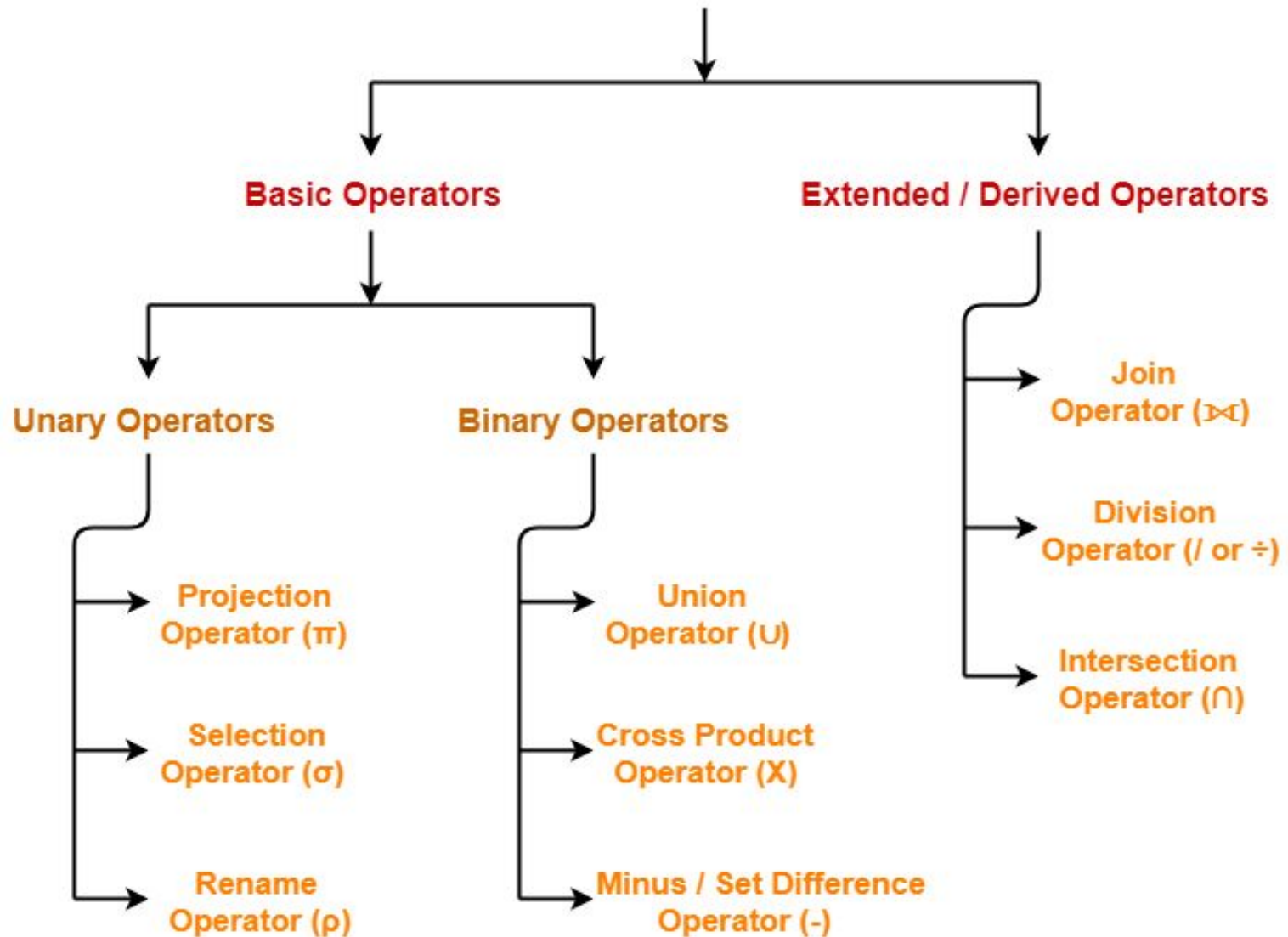


Chapter 4: Formal Relational Query Languages

Outline

- Relational Algebra
- Tuple Relational Calculus
- Domain Relational Calculus

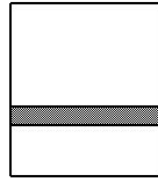
Relational Algebra Operators



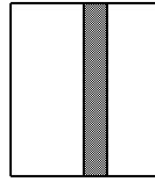
Relational Algebra

- Procedural language
- Six basic operators
 - select: σ
 - project: π
 - union: \cup
 - set difference: $-$
 - Cartesian product: \times
 - rename: ρ
- The operators take **one or two relations as inputs and produce a new relation as a result.**

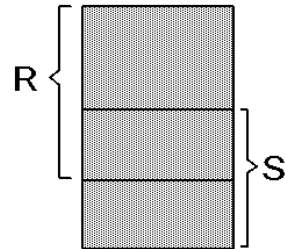
Select



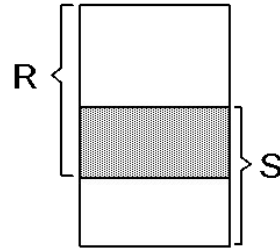
Project



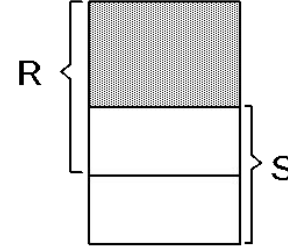
Union



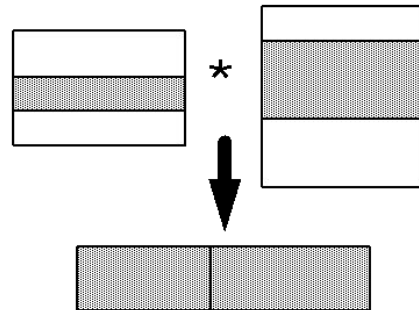
Intersection



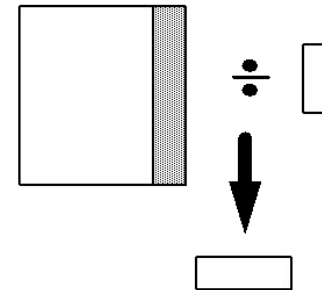
Difference



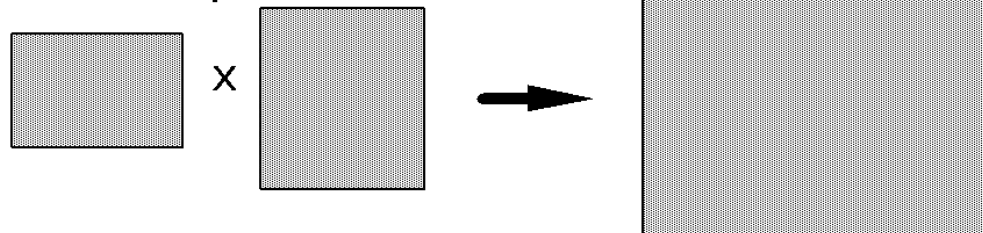
Join



Division



Cartesian product



Select Operation

- Notation: $\sigma_p(r)$
- p is called the **selection predicate**
- Defined as:

$$\sigma_p(r) = \{t \mid t \in r \text{ and } p(t)\}$$

Where p is a formula in propositional calculus consisting of **terms** connected by : \wedge (**and**), \vee (**or**), \neg (**not**)

Each **term** is one of:

<attribute> op <attribute> or <constant>

where op is one of: $=, \neq, >, \geq, <, \leq$

- Example of selection:

$$\sigma_{dept_name="Physics"}(instructor)$$

More Examples

- $\sigma_{\text{subject}=\text{"database"}}(\text{Books})$

- Selects tuples from books where subject is 'database'.

- $\sigma_{\text{subject}=\text{"database"} \text{ and } \text{Price}=\text{"450"}}(\text{Books})$

- Selects tuples from books where subject is 'database' and 'price' is 450.

- Selects tuples from books where subject is 'database' and 'price' is 450 or those books published after 2010.

- $\sigma_{\text{subject}=\text{"database"} \text{ and } \text{Price}=\text{"450"} \text{ or } \text{year} > \text{"2010"}}(\text{Books})$

Persons

Display record of person where the Age Is below 22.

$\sigma_{Age=22}(Persons)$

select only those rows or records which either have age equal to 22 or hometown equal to Texas.

$\sigma_{Hometown = 'Texas' OR Age=22}(Persons)$

Id	Name	Hometown	Age
10	Muhammad Ali	Ohio	29
11	Noah	Texas	25
12	Emma	Seattle	20
13	Arslan Ash	Lahore	22
14	Adam	Texas	18
15	Marques	California	22
16	Linus	Toronto	22

Project Operation

- Notation:

$$\Pi_{A_1, A_2, \dots, A_k}(r)$$

where A_1, A_2 are attribute names and r is a relation name.

- The result is defined as the relation of k columns obtained by erasing the columns that are not listed
- Duplicate rows removed from result, since relations are sets
- Example: To eliminate the *dept_name* attribute of *instructor*

$$\Pi_{ID, name, salary}(instructor)$$

Regno	Branch	Section
1	CSE	A
2	ECE	B
3	CIVIL	B
4	IT	A

- display regno column of student table
 - $\Pi_{Regno}(Student)$
- To display branch, section column of student table
 - $\Pi_{branch,section}(Student)$
- To display regno, section of ECE students
 - $\Pi_{regno,section}(\sigma_{Branch="ECE"}(Student))$

Cust

<u>cid</u>	cname	rating	salary
38	R. Rudy	9	95
32	G. Grumpy	8	55
51	S. Sneazy	5	95
78	R. Rusty	10	55

Display customer name and rating whose rating is greater than 7.

$\Pi_{cname, rating}(\sigma_{rating > 7}(Cust))$

Union Operation

- Notation: $r \cup s$

- Defined as:

$$r \cup s = \{t \mid t \in r \text{ or } t \in s\}$$

- For $r \cup s$ to be valid.

1. r, s must have the *same arity* (same number of attributes)
2. The attribute domains must be **compatible** (example: 2nd column of r deals with the same type of values as does the 2nd column of s)

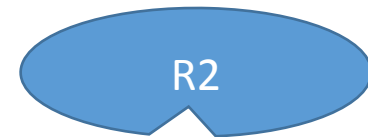
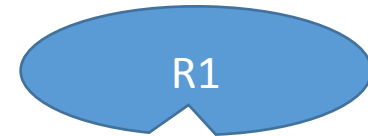
- Example: **to find all courses taught in the Fall 2009 semester, or in the Spring 2010 semester, or in both**

$$\Pi_{course_id} (\sigma_{semester="Fall" \wedge year=2009} (section)) \cup$$

$$\Pi_{course_id} (\sigma_{semester="Spring" \wedge year=2010} (section))$$

Regno	Branch	Section
1	CSE	A
2	ECE	B
3	MECH	B
4	CIVIL	A
5	CSE	B

Regno	Branch	Section
1	CIVIL	A
2	CSE	A
3	ECE	B



Display all the regno of R1 and R2 $\Pi_{Regno} (R1) \cup \Pi_{Regno} (R2)$

To retrieve branch and section of student from table R1 and R2

$\Pi_{Branch,Section} (R1) \cup \Pi_{Branc,Section} (R2)$

Table 1: COURSE

Course_Id	Student_Name	Student_Id
-----	-----	-----
C101	Aditya	S901
C104	Aditya	S901
C106	Steve	S911
C109	Paul	S921
C115	Lucy	S931

Table 2: STUDENT

Student_Id	Student_Name	Student_Age
-----	-----	-----
S901	Aditya	19
S911	Steve	18
S921	Paul	19
S931	Lucy	17
S941	Carl	16
S951	Rick	18

Display Student_Id of that student whose name is Aditya and whose age is 19.

Display Student_Name whose age is greater than 17 or obtain course C101 or C104.

Set Difference Operation

- Notation $r - s$

- Defined as:

$$r - s = \{t \mid t \in r \text{ and } t \notin s\}$$

- Set differences must be taken between **compatible** relations.
 - r and s must have the **same** arity
 - attribute domains of r and s must be compatible
- Example: to find all courses taught in the Fall 2009 semester, but not in the Spring 2010 semester

$$\Pi_{course_id} (\sigma_{semester="Fall" \wedge year=2009} (section)) - \Pi_{course_id} (\sigma_{semester="Spring" \wedge year=2010} (section))$$

STU_INDOOR	Regno	Sport
3 records	BIT001	Chess
	BIT023	Carrom
	BCE020	Badminton

STU_OUTDOOR	Regno	Sport
3 records	BIT001	Soccer
	BIT023	Soccer
	BME023	Cricket

Find the regno of students who plays indoor sports but not outdoor sports.

$$(\Pi_{regno} (stu_indoor)) - (\Pi_{regno} (stu_outdoor))$$

Find the regno of students who plays outdoor sports but not indoor sports.

$$(\Pi_{regno} (stu_outdoor)) - (\Pi_{regno} (stu_indoor))$$

STU_INDOOR	Regno	Sport
4 records	BIT001	Chess
	BIT023	Carrom
	BCE020	Badminton
	BEE001	Badminton

STU_OUTDOOR	Regno	Sport
3 records	BIT001	Soccer
	BIT023	Soccer
	BME023	Cricket

Find the regno of students who plays only badminton as indoor sports but not any of the outdoor sports.

SQL: *(SELECT regno FROM stu_indoor WHERE sport='Badminton') MINUS (SELECT regno FROM stu_outdoor);*

$(\Pi_{regno} (\sigma_{sport = 'Badminton'}(stu_indoor))) - (\Pi_{regno} (stu_outdoor))$

Set-Intersection Operation

- Notation: $r \cap s$
- Defined as:
 - $r \cap s = \{ t \mid t \in r \textbf{ and } t \in s \}$
- Assume:
 - r, s have the *same arity*
 - attributes of r and s are compatible
- Note: $r \cap s = r - (r - s)$

Depositor

ID	Name
1	A
2	B
3	C

Borrower

ID	Name
2	B
3	A
5	D

Find all the customers whose account is in the bank and have taken out a loan.

$\Pi_{\text{Name}}(\text{Depositor}) \cap \Pi_{\text{Name}}(\text{Borrower})$

STU_INDOOR	Regno	Sport
4 records	BIT001	Chess
	BIT023	Carrom
	BCE020	Badminton
	BEE001	Badminton

STU_OUTDOOR	Regno	Sport
3 records	BIT001	Soccer
	BIT023	Soccer
	BCE020	Cricket

Find the regno of students who plays both badminton and any of the outdoor sports.

SQL: *(SELECT regno FROM stu_indoor WHERE sport='Badminton')*
INTERSECT (SELECT regno FROM stu_outdoor);

RA: $(\Pi_{\text{regno}} (\sigma_{\text{sport} = \text{'Badminton'}}(\text{stu_indoor}))) \cap (\Pi_{\text{regno}} (\text{stu_outdoor}))$

Cartesian-Product Operation

- Notation $r \times s$
- Defined as:

$$r \times s = \{t \ q \mid t \in r \textbf{ and } q \in s\}$$

- Assume that attributes of $r(R)$ and $s(S)$ are disjoint. (That is, $R \cap S = \emptyset$).
- If attributes of $r(R)$ and $s(S)$ are not disjoint, then renaming must be used.

SNO	FNAME	LNAME
1	Albert	Singh
2	Nora	Fatehi

ROLLNO	AGE
5	18
9	21

Student x Details

SNO	FNAME	LNAME	ROLLNO	AGE
1	Albert	Singh	5	18
1	Albert	Singh	9	21
2	Nora	Fatehi	5	18
2	Nora	Fatehi	9	21

Rename Operation

- The RENAME operation is used to rename the output of a relation.
 - We may **want to save the result** of a relational algebra expression as a relation so that **we can use it later**.
 - We may want to join a relation with itself, in that case, it becomes too confusing to specify which one of the tables we are talking about, in that case, we rename one of the tables and perform join operations on them.
- Example:

$$\rho_x(E)$$

returns the expression E under the name X

- If a relational-algebra expression E has arity n , then

$$\rho_{x(A_1, A_2, \dots, A_n)}(E)$$

returns the result of expression E under the name X , and with the attributes renamed to A_1, A_2, \dots, A_n .

- Query to rename the relation Student as Male Student and the attributes of Student – RollNo, SName as (Sno, Name) where rollno greater than 20

$\rho_{\text{MaleStudent(Sno, Name)}} \pi_{\text{RollNo, SName}} (\sigma_{\text{RollNo} > 20} (\text{Student}))$

Student

Id	Name	Contact	Email ID	City	Faculty Id
1	Rakesh	9889789877	rakesh@gmail.com	Anand	2
2	Ramesh	6545654565	Ramesh@hotmail.com	Baroda	2
3	Krishna	8454814584	Krishna@gmail.com	Anand	1
4	Himani	7845847564	himani@yahoo.co.in	Surat	2
5	Tejas	8456845685	tej@hotmail.com	Surat	3

Faculty

ID	F_name	Subject_id	Subject
1	Sandeep	CE120	DBMS
2	Deepak	CE121	DSA
3	Ankita	CE425	SOC
4	Mitali	CE564	CC

- Find the student list who belongs to surat city
- Find the faculty name who belongs to “DBMS” Subject
- Find student name who belongs to city to “Anand” and who’s Faculty id is =”1”
- Find faculty name of those student who belongs to city=”Anand”

Student

Id	Student_Name	Semester	Skills
1	Rajesh	Fall	ML
2	Rakesh	Spring	ML
3	Hema	Fall	Spring
4	Jignesh	Fall	ML
6	Bhavesb	Spring	Spring

Programming

Id	Student_Name	Programming Language
1	Rakesh	Python
2	Jignesh	Java
5	Mihir	C#
3	Hema	Java

- Find Student name whose skill is in “ML” and also programming language as “Python”
- Find those student name who belongs to semester “Fall” but not having Programming language as “Java”
- Make a table as Student_Participation with the column as Name_of_Student find details of those student who belongs to semester “Fall” and skill as “ML” also those who good in “Java”

Tuple Relational Calculus

Tuple Relational Calculus

- A **nonprocedural query language**, where each query is of the form

$$\{t \mid P(t)\}$$

- It is the set of **all tuples t such that predicate P is true for t**
- t is a *tuple variable*, $t[A]$ denotes the value of tuple t on **attribute A**
- $t \in r$ denotes that tuple t is in relation r
- **P is a formula** similar to that of the predicate calculus

Predicate Calculus Formula

1. Set of attributes and constants
2. Set of comparison operators: (e.g., $<$, \leq , $=$, \neq , $>$, \geq)
3. Set of connectives: and (\wedge), or (\vee), not (\neg)
4. Implication (\Rightarrow): $x \Rightarrow y$, if x is true, then y is true
5. Set of quantifiers:
 - ▶ $\exists t \in r (Q(t)) \equiv$ "there exists" a tuple t in relation r such that **predicate $Q(t)$** is true
 - ▶ $\forall t \in r (Q(t)) \equiv Q$ is true "**for all**" tuples t in relation r

FirstName	LastName	Age
Pinky	Singh	30
Dolly	Singh	31
Tannu	Pratap	28
Lucky	Maheswari	27

Write Non Procedural Query

Display the last name of those students where age is greater than 30.

{ t.LastName | Student(t) and t.Age > 30 }

Display name of student where last name is "singh"

{ t.FirstName | Student(t) and t.LastName="Singh" }

Example Queries

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Figure 6.1 The *instructor* relation.

- Find the *ID*, *name*, *dept_name*, *salary* for instructors whose salary is greater than \$80,000

$$\{t \mid t \in \text{instructor} \wedge t[\text{salary}] > 80000\}$$

Notice that a relation on schema (*ID*, *name*, *dept_name*, *salary*) is implicitly defined by the query

Convert Non Procedural Query in Quantifiers

As in the previous query, but output only the *ID* attribute value

$$\{t \mid \exists s \in \text{instructor} (t[\text{ID}] = s[\text{ID}] \wedge s[\text{salary}] > 80000)\}$$

Notice that a relation on schema (*ID*) is implicitly defined by the query

<i>dept_name</i>	<i>building</i>	<i>budget</i>
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Electrical Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000
<i>null</i>	Painter	<i>null</i>

department

- Find the names of all instructors whose department is in the Watson building

$$\{t \mid \exists s \in \text{instructor} (t[\text{name}] = s[\text{name}] \wedge \exists u \in \text{department} (u[\text{dept_name}] = s[\text{dept_name}] \wedge u[\text{building}] = \text{"Watson"}))\}$$

<i>course_id</i>	<i>sec_id</i>	<i>semester</i>	<i>year</i>	<i>building</i>	<i>room_number</i>	<i>time_slot_id</i>
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A
CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C
FIN-201	1	Spring	2010	Packard	101	B
HIS-351	1	Spring	2010	Painter	514	C
MU-199	1	Spring	2010	Packard	101	D
PHY-101	1	Fall	2009	Watson	100	A

Figure 2.6 The *section* relation.

Example Queries

- Find the set of all `course_id` taught in the Fall 2009 semester, or in the Spring 2010 semester, or both

$$\{t \mid \exists s \in \text{section } (t [\text{course_id}] = s [\text{course_id}] \wedge s [\text{semester}] = \text{"Fall"} \wedge s [\text{year}] = 2009 \\ \vee \exists u \in \text{section } (t [\text{course_id}] = u [\text{course_id}] \wedge u [\text{semester}] = \text{"Spring"} \wedge u [\text{year}] = 2010)\}$$

Example Quantifiers Queries

- Find the set of all `course_id` taught in the Fall 2009 semester, and in the Spring 2010 semester

$$\{t \mid \exists s \in \text{section } (t [\text{course_id}] = s [\text{course_id}] \wedge s [\text{semester}] = \text{"Fall"} \wedge s [\text{year}] = 2009 \wedge \exists u \in \text{section } (t [\text{course_id}] = u [\text{course_id}] \wedge u [\text{semester}] = \text{"Spring"} \wedge u [\text{year}] = 2010))\}$$

- Find the set of all `course_id` taught in the Fall 2009 semester, but not in the Spring 2010 semester

$$\{t \mid \exists s \in \text{section } (t [\text{course_id}] = s [\text{course_id}] \wedge s [\text{semester}] = \text{"Fall"} \wedge s [\text{year}] = 2009 \wedge \neg \exists u \in \text{section } (t [\text{course_id}] = u [\text{course_id}] \wedge u [\text{semester}] = \text{"Spring"} \wedge u [\text{year}] = 2010))\}$$

Universal Quantification

- Find all students who have taken all courses offered in the Biology department
 - $\{t \mid \exists r \in \text{student} (t[ID] = r[ID]) \wedge$
 $(\forall u \in \text{course} (u[\text{dept_name}] = \text{"Biology"} \Rightarrow$
 $\exists s \in \text{takes} (t[ID] = s[ID] \wedge$
 $s[\text{course_id}] = u[\text{course_id}]))\}$

Domain Relational Calculus

Domain Relational Calculus

- A nonprocedural query language equivalent in power to the tuple relational calculus
- Each query is an expression of the form:

$$\{ \langle x_1, x_2, \dots, x_n \rangle \mid P(x_1, x_2, \dots, x_n) \}$$

- x_1, x_2, \dots, x_n represent domain variables
- P represents a formula similar to that of the predicate calculus

Example Queries

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Figure 6.1 The *instructor* relation.

- Find the *ID*, *name*, *dept_name*, *salary* for instructors whose salary is greater than \$80,000
 - $\{ \langle i, n, d, s \rangle \mid \langle i, n, d, s \rangle \in \text{instructor} \wedge s > 80000 \}$
- As in the previous query, but output only the *ID* attribute value
 - $\{ \langle i \rangle \mid \langle i, n, d, s \rangle \in \text{instructor} \wedge s > 80000 \}$

<i>dept_name</i>	<i>building</i>	<i>budget</i>
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Electrical Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000
<i>null</i>	Painter	<i>null</i>

department

Find the names of all instructors whose department is in the Watson building

$$\{ \langle n \rangle \mid \exists i, d, s (\langle i, n, d, s \rangle \in \text{instructor} \wedge \exists b, a (\langle d, b, a \rangle \in \text{department} \wedge b = \text{"Watson"})) \}$$

<i>course_id</i>	<i>sec_id</i>	<i>semester</i>	<i>year</i>	<i>building</i>	<i>room_number</i>	<i>time_slot_id</i>
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A
CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C
FIN-201	1	Spring	2010	Packard	101	B
HIS-351	1	Spring	2010	Painter	514	C
MU-199	1	Spring	2010	Packard	101	D
PHY-101	1	Fall	2009	Watson	100	A

Figure 2.6 The *section* relation.

Example Queries

- Find the set of all courses taught in the Fall 2009 semester, or in the Spring 2010 semester, or both

$$\{ \langle c \rangle \mid \exists a, s, y, b, r, t (\langle c, a, s, y, b, r, t \rangle \in \text{section} \wedge s = \text{"Fall"} \wedge y = 2009) \vee \exists a, s, y, b, r, t (\langle c, a, s, y, b, r, t \rangle \in \text{section}] \wedge s = \text{"Spring"} \wedge y = 2010) \}$$

This case can also be written as

$$\{ \langle c \rangle \mid \exists a, s, y, b, r, t (\langle c, a, s, y, b, r, t \rangle \in \text{section} \wedge ((s = \text{"Fall"} \wedge y = 2009) \vee (s = \text{"Spring"} \wedge y = 2010))) \}$$

- Find the set of all courses taught in the Fall 2009 semester, and in the Spring 2010 semester

$$\{ \langle c \rangle \mid \exists a, s, y, b, r, t (\langle c, a, s, y, b, r, t \rangle \in \text{section} \wedge s = \text{"Fall"} \wedge y = 2009) \wedge \exists a, s, y, b, r, t (\langle c, a, s, y, b, r, t \rangle \in \text{section}] \wedge s = \text{"Spring"} \wedge y = 2010) \}$$

End of Chapter 4


GATE Question


Multiple Answer Question

- Consider the following three relations in a relational database.

Employee(eId,Name),Brand(bId,bName),Own(eId,bId)

- Which of the following relational algebra expressions return the set of eIds who own all the brands?

A. $\Pi_{eId} (\Pi_{eId,bId}(Own) / \Pi_{bId}(Brand))$ 

B. $\Pi_{eId}(Own) - \Pi_{eId} ((\Pi_{eId}(Own) \times \Pi_{bId}(Brand)) - \Pi_{eId,bId}(Own))$ 

C. $\Pi_{eId} (\Pi_{eId,bId}(Own) / \Pi_{bId}(Own))$


D. $\Pi_{eId} ((\Pi_{eId}(Own) \times \Pi_{bId}(Own)) / \Pi_{bId}(Brand))$

- The following relation records the age of 500 employees of a company, where empNo (indicating the employee number) is the key:
empAge(empNo,age)

Consider the following relational algebra expression:

$$\Pi_{\text{empNo}}(\text{empAge} \bowtie_{(\text{age} > \text{age1})} \rho_{\text{empNo1, age1}}(\text{empAge}))$$

- What does the above expression generate?

- A) Employee numbers of only those employees whose age is the maximum
- B) Employee numbers of only those employees whose age is more than the age of exactly one other employee
- C) Employee numbers of all employees whose age is not the minimum 
- D) Employee numbers of all employees whose age is the minimum

Consider the following relations $P(X, Y, Z)$, $Q(X, Y, T)$ and $R(Y, V)$.

Table: P		
X	Y	Z
X1	Y1	Z1
X1	Y1	Z2
X2	Y2	Z2
X2	Y4	Z4


Table: Q		
X	Y	T
X2	Y1	2
X1	Y2	5
X1	Y1	6
X3	Y3	1

Table: R	
Y	V
Y1	V1
Y3	V2
Y2	V3
Y2	V2

How many tuples will be returned by the following relational algebra query?

$$\Pi_x(\sigma_{(P.Y=R.Y \wedge R.V=V2)}(P \times R)) - \Pi_x(\sigma_{(Q.Y=R.Y \wedge Q.T>2)}(Q \times R))$$

Answer: _____

- A) 1 
- B) 2
- C) 3
- D) 4

Solution

$R \cdot V = V2$, there are two tuples which have Y parameter as $Y3$ and $Y2$.

$P \cdot Y = R \cdot Y$, there are no coincide with $Y3$, and there is one tuple coincide with $Y2$ which have X parameter as $X2$.

$$\Pi_X(\sigma_{(P.Y=R.Y \wedge R.V=V2)}(P \times R)) = \{X_2\}$$

$Q \cdot T > 2$, there are two tuples which have Y parameter as $Y1$ and $Y2$ which have X parameter as $X1$.

check atleast one tuple is matched $Q \cdot Y = R \cdot Y$ in this query, that's enough to result $X1$.

$$\Pi_X(\sigma_{(Q.Y=R.Y \wedge Q.T>2)}(Q \times R)) = \{X_1\}$$

$$\Pi_X(\sigma_{(P.Y=R.Y \wedge R.V=V2)}(P \times R)) - \Pi_X(\sigma_{(Q.Y=R.Y \wedge Q.T>2)}(Q \times R)) = \{X_2\} - \{X_1\} = \{X_2\}$$

Number of Tuples = 1

Consider the relations $r(A,B)$ and $s(B,C)$, where $s.B$ is a primary key and $r.B$ is a foreign key referencing $s.B$. Consider the query


$$Q: r \bowtie (\sigma_{B < 5}(s))$$

Let LOJ denote the natural left outer-join operation. Assume that r and s contain no null values.

Which of the following is NOT equivalent to Q ?

A. $\sigma_{B < 5}(r \bowtie s)$

B. $\sigma_{B < 5}(r \text{ } LOJ \text{ } s)$

C. $r \text{ } LOJ \text{ } (\sigma_{B < 5}(s))$ 

D. $\sigma_{B < 5}(r) \text{ } LOJ \text{ } s$

Solutions:

Option a,b,d will restrict all record with $B < 5$ but option C will include record with $b \geq 5$ also, so false.

C is the answer.

- Consider the relational schema given below, where **eld** of the relation **dependent** is a foreign key referring to **empld** of the relation **employee**. Assume that every employee has at least one associated dependent in the **dependent** relation.

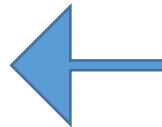
- employee** (empld, empName, empAge)
- dependent** (depld, eld, depName, depAge)

- Consider the following relational algebra query:

$$\Pi_{\text{empld}}(\text{employee}) - \Pi_{\text{empld}}(\text{employee} \bowtie_{(\text{empld}=\text{eld})} \wedge_{(\text{empAge} \leq \text{depAge})} \text{dependent})$$

- The above query evaluates to the set of **emplds** of employees whose age is greater than that of

- A) some dependent.
- B) all dependents.
- C) some of his/her dependents.
- D) all of his/her dependents.



Consider the following relations A , B and C :


A		
ID	Name	Age
12	Arun	60
15	Shreya	24
99	Rohit	11

B		
ID	Name	Age
15	Shreya	24
25	Hari	40
98	Rohit	20
99	Rohit	11

C		
ID	Phone	Area
10	2200	02
99	2100	01


How many tuples does the result of the following relational algebra expression contain? Assume that the schema of $A \cup B$ is the same as that of A .

$$(A \cup B) \bowtie_{A.Id > 40 \vee C.Id < 15} C$$

- A. 7 
- B. 4
- C. 5
- D. 9

- Consider the following relation schemas :
 - b-Schema = (b-name, b-city, assets)
 - a-Schema = (a-num, b-name, bal)
 - d-Schema = (c-name, a-number)
- Let branch, account and depositor be respectively instances of the above schemas. Assume that account and depositor relations are much bigger than the branch relation.
- Consider the following query:


$$\Pi_{c\text{-name}} (\sigma_{b\text{-city} = \text{"Agra"} \wedge \text{bal} < 0} (\text{branch} \bowtie (\text{account} \bowtie \text{depositor})))$$
- Which one of the following queries is the most efficient version of the above query ?

- A) $\Pi_{c\text{-name}} (\sigma_{\text{bal} < 0} (\sigma_{b\text{-city} = \text{"Agra"}} \text{branch} \bowtie \text{account}) \bowtie \text{depositor})$
- B) $\Pi_{c\text{-name}} (\sigma_{b\text{-city} = \text{"Agra"}} \text{branch} \bowtie (\sigma_{\text{bal} < 0} \text{account} \bowtie \text{depositor}))$
- C) $\Pi_{c\text{-name}} ((\sigma_{b\text{-city} = \text{"Agra"}} \text{branch} \bowtie \sigma_{b\text{-city} = \text{"Agra"} \wedge \text{bal} < 0} \text{account}) \bowtie \text{depositor})$
- D) $\Pi_{c\text{-name}} (\sigma_{b\text{-city} = \text{"Agra"}} \text{branch} \bowtie (\sigma_{b\text{-city} = \text{"Agra"} \wedge \text{bal} < 0} \text{account} \bowtie \text{depositor}))$
- 

Information about a collection of students is given by the relation **studInfo**(studId, name, sex).

The relation **enroll**(studId, courseId) gives which student has enrolled for (or taken) what course(s). Assume that every course is taken by at least one male and at least one female student. What does the following relational algebra expression represent?

$$\Pi_{\text{courseId}} \left(\left(\Pi_{\text{studId}} \left(\sigma_{\text{sex}=\text{"female"}} (\text{studInfo}) \right) \times \Pi_{\text{courseId}} (\text{enroll}) \right) - \text{enroll} \right)$$

- A) Courses in which all the female students are enrolled.
- B) Courses in which a proper subset of female students are enrolled. 
- C) Courses in which only male students are enrolled.
- D) None of the above

Consider the relation Student (name, sex, marks), where the primary key is shown underlined, pertaining to students in a class that has at least one boy and one girl. What does the following relational algebra expression produce? (Note: ρ is the rename operator).

$$\pi_{name}\{\sigma_{sex=female}(Student)\} - \pi_{name}(Student \bowtie_{(sex=female \wedge x=marks \leq m)} \rho_{n,x,m}(Student))$$

- A) names of girl students with the highest marks
- B) names of girl students with more marks than some boy student
- C) names of girl students with marks not less than some boy student
- D) names of girl students with more marks than all the boy students



Solution

$$\pi_{name} \{ r_{sex=female}(Student) \} - \pi_{name} (Student_{(sex=female \wedge x=male \wedge marks \leq m)} \rho_{n,x,m}(Student))$$

lets try to break the query into parts and then resolve

Lets take the second part :

$$\pi_{name} (Student_{(sex=female \wedge x=male \wedge marks \leq m)} \rho_{n,x,m}(Student))$$

Here, cross product of two copies of Student relation is carried out. To effectively write the selection criteria, one relation of student is renamed with new attributes as name to n, sex to x and marks to m.

Now after having taking the cross product, selection operation is performed on the result based on below criteria :

$$(sex=female \wedge x=male \wedge marks \leq m)$$

which selects all such females whose marks are less or equal to atleast one male student (atleast one male student M1 should exist whose marks must be either greater or equal to the marks of Female student F1 for the record to be selected by the selection statement) .

Now, if we look at the first part,

$$\pi_{name} \{ r_{sex=female}(Student) \}$$

Solution

This selects all female students from the relation.

Now if we look the original problem, it is more or like A-B

$$\pi_{name}\{\sigma_{sex=female}(Student)\} - \pi_{name}(Student_{(sex=female \wedge x=marks \leq m)} \rho_{n,x,m}(Student))$$

A - B = A intersection B'

If we consider the second part of the problem as B which gives all female students which have marks either less or equal to atleast one male student

complement of this statement would be

all female students which have marks more than all the male students.

And this if taken intersection with all female students

will give all female students with marks more than all male students

So, Option (d) is the answer.